01 Introduction to C# and Data Types

- Understanding Data Types

1. What type would you choose for the following “numbers”?
   A person’s telephone number - string
   A person’s height - string
   A person’s age - int
   A person’s gender (Male, Female, Prefer Not To Answer) - enum
   A person’s salary - decimal
   A book’s ISBN - string
   A book’s price - decimal
   A book’s shipping weight - double
   A country’s population - uint
   The number of stars in the universe - ulong
   The number of employees in each of the small or medium businesses in the
   United Kingdom (up to about 50,000 employees per business) - short
2. What are the difference between value type and reference type variables? What is
   boxing and unboxing?
   a. value type will directly hold the value, while reference type will hold the memory address/reference for this value
   b. value types are stored in stack memory, while reference types will be stored in heap memory
   c. Value type will not be collected by garbage collector, while reference type will be collected by garbage collector
   d. value type can be created by Struct or Enum, Reference type can be created by Class, Interface, Delegate or Array etc
   e. value type cannot accept null values, but reference type can accept null values

   conversion of a value type to reference type -- boxing
   conversion of a reference type to a value type -- unboxing
3. What is meant by the terms managed resource and unmanaged resource in .NET

   Managed resource basically means "managed memory" that is managed by the garbage collector. When we no longer have any references to a managed object (which uses managed memory), the garbage collector will (eventually) release that memory for us.

   Unmanaged resources are then everything that the garbage collector does not know about, such as such as files, stream and handles, so we need to work with IDisposable interface to release.
4. What's the purpose of Garbage Collector in .NET?

   Garbage collector (GC) serves as an automatic memory manager, which manages the allocation and release of memory for an application.

- Practice number sizes and rages

1. The minimum and maximum values they can have: sbyte, byte, short, ushort, int, uint, long,
   ulong, float, double, and decimal.

```
Console.WriteLine($"Number of bytes for int is {sizeof(sbyte)}");
Console.WriteLine($"Minimum value for in is {sbyte.MinValue}");
Console.WriteLine($"Maximum value for in is {sbyte.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(byte)}");
Console.WriteLine($"Minimum value for in is {byte.MinValue}");
Console.WriteLine($"Maximum value for in is {byte.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(short)}");
Console.WriteLine($"Minimum value for in is {short.MinValue}");
Console.WriteLine($"Maximum value for in is {short.MaxValue}");
Console.WriteLine($"Number of bytes for int is {sizeof(ushort)}");
Console.WriteLine($"Minimum value for in is {ushort.MinValue}");
Console.WriteLine($"Maximum value for in is {ushort.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(int)}");
Console.WriteLine($"Minimum value for in is {int.MinValue}");
Console.WriteLine($"Maximum value for in is {int.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(uint)}");
Console.WriteLine($"Minimum value for in is {uint.MinValue}");
Console.WriteLine($"Maximum value for in is {uint.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(long)}");
Console.WriteLine($"Minimum value for in is {long.MinValue}");
Console.WriteLine($"Maximum value for in is {long.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(ulong)}");
Console.WriteLine($"Minimum value for in is {ulong.MinValue}");
Console.WriteLine($"Maximum value for in is {ulong.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(float)}");
Console.WriteLine($"Minimum value for in is {float.MinValue}");
Console.WriteLine($"Maximum value for in is {float.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(double)}");
Console.WriteLine($"Minimum value for in is {double.MinValue}");
Console.WriteLine($"Maximum value for in is {double.MaxValue}");

Console.WriteLine($"Number of bytes for int is {sizeof(decimal)}");
Console.WriteLine($"Minimum value for in is {decimal.MinValue}");
Console.WriteLine($"Maximum value for in is {decimal.MaxValue}");
```

2. Write program to enter an integer number of centuries and convert it to years, days, hours, minutes, seconds, milliseconds, microseconds, nanoseconds. Use an appropriate data type for every data conversion. Beware of overflows!

```
public void ConvertDate(uint century)
        {
            uint year = century * 100;
            uint days = year * 365 + 24 * century;
            uint hours = days * 24;
            uint minutes = hours * 60;
            uint seconds = minutes * 60;
            ulong milliseconds = Convert.ToUInt64(seconds) * 1000;
            ulong microseconds = milliseconds * 1000;
            ulong nanoseconds = microseconds * 1000;
            Console.WriteLine($"{century} centuries = {year} years = {days} days = {hours} hours = {minutes} minutes = {seconds} second
                $"= {milliseconds} miliseconds = {microseconds} microseconds = {nanoseconds} nanoseconds");
        }
```

- Controlling Flow and Converting Types
  1. What happens when you divide an int variable by 0?

     Compile time error: Division by constant 0
  2. What happens when you divide a double variable by 0?

     double.PositiveInfinity
  3. What happens when you overflow an int variable, that is, set it to a value beyond its range?

     Compile time error: cannot explicitly convert int type to other data types
  4. What is the difference between x = y++; and x = ++y;?

     ```
     x = y++;
     //equivalent to:
     x = y;
     y++;

     x = ++y;
     ```

```
//equivalent to:
y++;
x = y;
```

5. What is the difference between break, continue, and return when used inside a loop statement?

   break: jump out of the current loop

   continue: skip the current iteration, continue with the following iteration

   return: terminate the loop with some return the control to the calling method

6. What are the three parts of a for statement and which of them are required?

   - An initializer expression, which executes once at the start of the loop;
   - A conditional expression, which executes on every iteration at the start of the loop to check whether the looping should continue; -required
   - An iterator expression, which executes on every loop at the bottom of the statement.

7. What is the difference between the = and == operators?

   = operator: assign the value to the left hand side object

   == operator: check if two objects has the same value

8. Does the following statement compile? for (; true;) ;

   Yes

9. What does the underscore _ represent in a switch expression?

   The underscore character is used to represent the default return value.

10. What interface must an object implement to be enumerated over by using the foreach statement?

    IEnumerable

- Practice loops and operators

1. FizzBuzzis

```
public void FizzBuzz(int num)
{
    for (int i = 1; i <= num; i++)
    {
        if (i % 15 == 0)
        {
            Console.Write("FizzBuzz ");
        } else if (i % 3 == 0)
        {
            Console.Write("Fizz ");
        } else if (i % 5 == 0)
        {
            Console.Write("Buzz ");
        } else
        {
            Console.Write($"{i} ");
        }
    }
}
```

2. What will happen if this code executes?

```
int max = 500;
for (byte i = 0; i < max; i++)
{
    Console.WriteLine(i);
}
```

```
//infinite loop
//becuase once there is an overflow on byte, it will start over from 0, thus the loop will never end
// how to solve: inside for loop, we should use int i instead of byte i
```

3. Print-a-Pyramid.

```
public void PrintPyramid(int size)
{
    for (int i = 1; i <= size; i++)
    {
        for (int j = 0; j <= size - i; j++)
        {
            Console.Write(" ");
        }
        for (int j = 1; j < i * 2; j++)
        {
            Console.Write("*");
        }
        Console.WriteLine();
    }
}
```

4. Generates a random number between 1 and 3, and ask user to guess what the number is

```
public void GuessNumber()
{
    int correctNumber = new Random().Next(3) + 1;
    Console.Write("Please enter a number between 1 and 3 => ");
    int guessedNumber = Convert.ToInt32(Console.ReadLine());
    if (correctNumber == guessedNumber)
    {
        Console.WriteLine("You got the correct answer!");
    } else if (correctNumber < 1 || correctNumber > 3)
    {
        Console.WriteLine("You guess was outside of the range");
    } else if (correctNumber < guessedNumber)
    {
        Console.WriteLine("A little bit too high.");
    } else
    {
        Console.WriteLine("A little bit too low.");
    }
}
```

5. Write a simple program that defines a variable representing a birth date and calculates how many days old the person with that birth date is currently.

```
public void DaysOld()
{
    Console.Write("Please enter the year you were born => ");
    int year = Convert.ToInt32(Console.ReadLine());
    Console.Write("Please enter the month you were born => ");
    int month = Convert.ToInt32(Console.ReadLine());
    Console.Write("Please enter the year you were born => ");
    int day = Convert.ToInt32(Console.ReadLine());
    DateTime DateOfBirth = new DateTime(year, month, day);
    double diff = (DateTime.Today - DateOfBirth).TotalDays;
    Console.WriteLine($"You have been lived for {diff} days!");
    int daysToNextAnniversary = Convert.ToInt32(10000 - (diff % 10000));
    Console.WriteLine($"There are {daysToNextAnniversary} days left to your next Anniversary");
}
```

6. Write a program that greets the user using the appropriate greeting for the time of day.

```
public void Greetings()
{
    DateTime moment = DateTime.Now;
    int hour = moment.Hour;
    if (hour < 12)
    {
        Console.WriteLine("Good Morning!");
    } else if (hour >= 12 && hour < 17)
    {
        Console.WriteLine("Good Afternoon!");
    } else if (hour >= 17 && hour < 21)
    {
        Console.WriteLine("Good Evening!");
    } else
    {
        Console.WriteLine("Good Night!");
    }
}
```

7. Write a program that prints the result of counting up to 24 using four different increments. First, count by 1s, then by 2s, by 3s, and finally by 4s.

```
public void CountByIncrement(int target)
{
    for (int i = 1; i <= 4; i++)
    {
        int res = 0;
        while (res <= target)
        {
            Console.Write($"{res} ");
            res = res + i;
        }
        Console.WriteLine();
    }
}
```